Toward A Mathematical Understanding of the Malware Problem

Michael Stephen Fiske

Aemea Institute

HICSS 53
January 9, 2020

Table of contents

Importance of Understanding the Problem

Will Ohmer and Orville Wright. Wright Patterson AFB.

Inventor and great grandfather Will Ohmer used to say:

"To invent something, identify and understand the problem."

Limitations of Malware Detection

Up against fundamental limits in Computer Science:

- No computer algorithm can detect all malware. [1]
- NP hard problems (traveling salesman) help encrypt and hide the malware. [2]

[1] Fred Cohen. Computer Viruses Theory and Experiments. Computers and Security. 6(1) 22–35, Feb. 1987.

[2] Eric Filiol. Malicious Cryptology and Mathematics. Cryptography and Security in Computing. Chapter 2. Intech, March 7, 2012.

Register Machine Vulnerability

- After C, JAVA or Python program is compiled, a register machine is a computer that executes the compiled instructions.
- Register machines execute one instruction at-a-time.
- Register machine hardware uses branch instructions (jump).
- A 1 or 2 bit flip in an instruction subverts the program.
- One rogue branch instruction can jump to a malware routine.
- In some cases, malware detection code won't execute.

Flipping 2 bits in a C program instruction

```
int greater_than(int p1, int p2)
{ return (p1 > p2); }

int less_than(int p1, int p2)
{ return (p1 < p2); }

int main(int argc, char* argv[])
{
   int nums[4] = {6, 9, 7, 8};
   display_numbers(nums, 4);
   printf("\n");
   sort_pr(nums, 4, "less_than", less_than);
   sort_pr(nums, 4, "greater_than", greater_than);
   return 0;
}
```

```
~MacBook-Air:sort$  ./sort
6 9 7 8

6 7 8 9  address of instruction less_than
1 1 0 1 1 0 1 0 0 1 0 0 0 0 0 0

9 8 7 6  address of instruction greater_than
1 1 0 1 1 0 1 0 0 0 0 1 0 0 0 0
```

Motivating Stable Computation

- Typical programming languages (C, JAVA, Python) use conditional branching instructions.
- 75% to 80% of control flow instructions in register machines are conditional branch instructions. [3]
- A register machine program's purpose can be subverted because its behavior is not stable w.r.t. small changes.

[3] J. Hennessy and D. Patterson. Computer Architecture. 5th Edition, Morgan Kaufmann, 2012. Figure A.14

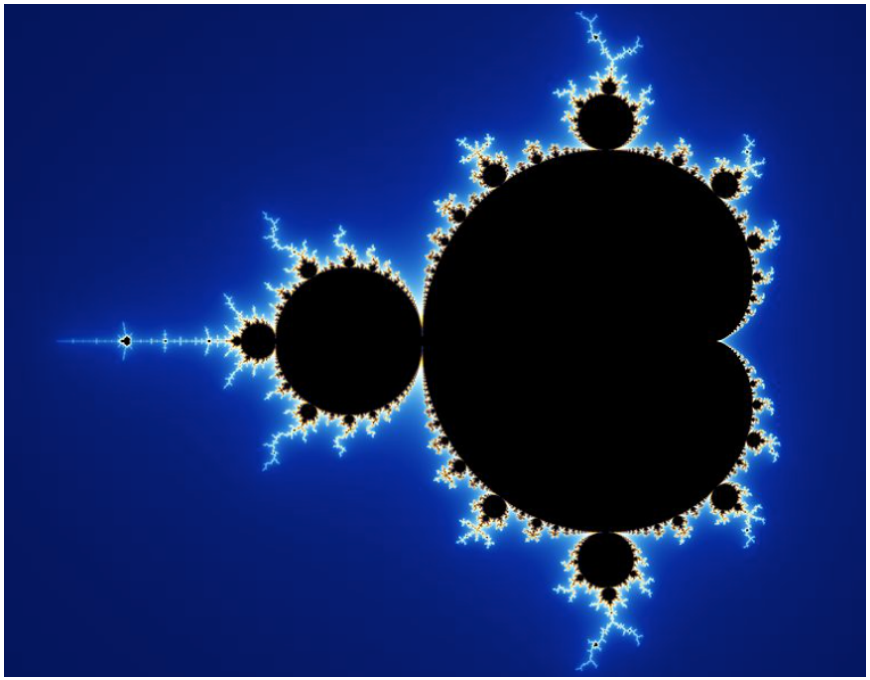
A Program is a Dynamical System

- Dynamical systems has studied stability for over 80 years. [4]
- Use mathematical tools (metric spaces, structural stability, topological conjugacy, ...) from dynamical systems theory.
- *Small changes* to a computer program can be measured as small changes to a dynamical system.
- Each Turing machine (computer program) maps to a finite set of affine maps (2×2 matrix + translation) in the x-y plane.

[4] Aleksandr Andronov and Lev Pontrjagin. Systèmes Grossiers. Dokl. Akad. Nauk., SSSR, 14, 247–251, 1937.

Mandelbrot Set is generated from a Dynamical System

Define function $f_c(z) = z^2 + c$. c is a complex number.



The Mandelbrot set is the set of all complex numbers c such that the orbit $0, \quad f_c(0), \quad f_c \circ f_c(0), \quad f_c \circ f_c \circ f_c(0), \quad \ldots$ is bounded.

ϕ Maps each Turing Instruction to a Unique Affine Map

Alphabet $A = \{\#, a, b\}$. Machine states $Q = \{q, r, s\}$.

Turing program η.

η	#	a	b
q	$(r, a, +1)$	$(h, b, +1)$	$(q, b, -1)$
r	$(q, b, -1)$	$(r, a, +1)$	$(r, b, +1)$
s	$(h, \#, +1)$	$(h, a, +1)$	$(h, b, +1)$

$$\eta(q, \#) = (r, a, +1) \quad \xrightarrow{\phi} \quad f_1(x, y) = (7x - 49,\ \tfrac{1}{7}y + 33)$$

$$\eta(r, \#) = (q, b, -1) \quad \xrightarrow{\phi} \quad f_2(x, y) = (\tfrac{1}{7}x + 16,\ 7y - 231)$$

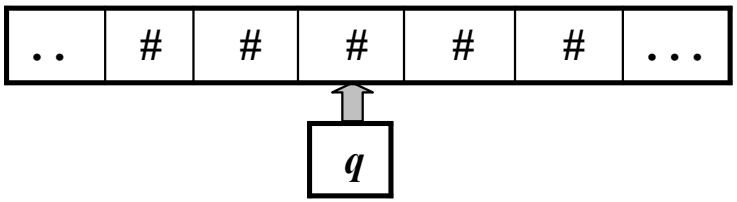
Turing Instruction Execution $\overset{\phi}{\leftrightarrow}$ Affine Map Iteration

$$\nu(h) = 0 \quad \nu(\#) = 1 \quad \nu(a) = 2 \quad \nu(b) = 3 \quad \nu(q) = 4 \quad \nu(r) = 5 \quad \nu(s) = 6$$

$$B = |Q| + |A| + 1 = 7$$

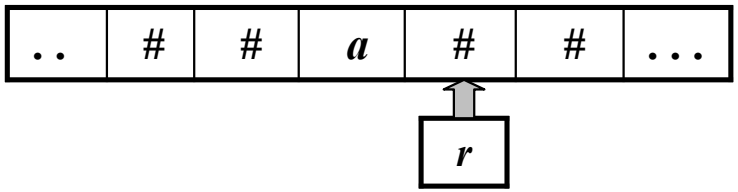
$$(q, k, T) \quad \overset{\phi}{\leftrightarrow} \quad \Big(\sum_{j=-1}^{\infty} \nu(T_{k+j+1}) B^{-j},\ B\nu(q) + \sum_{j=0}^{\infty} \nu(T_{k-j-1}) B^{-j} \Big)$$

Before machine execution starts



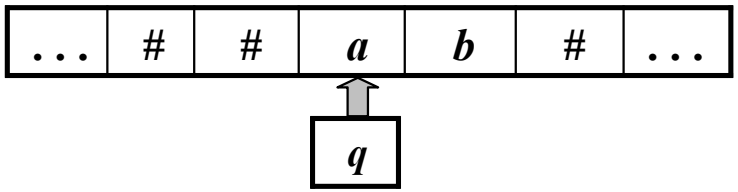
$$p_0 = \left(8\frac{1}{6}, 29\frac{1}{6}\right)$$

After Instruction $\eta(q, \#) = (r, a, +1)$



$$p_1 = f_1(p_0) = \left(8\frac{1}{6}, 37\frac{1}{6}\right)$$

After Instruction $\eta(r, \#) = (q, b, -1)$



$$p_2 = f_2(p_1) = \left(17\frac{1}{6}, 29\frac{1}{6}\right)$$

Topological Conjugacy and Structural Stability

- A topologically conjugacy h between two dynamical systems f, g means they have equivalent dynamics.
- h maps halting configurations to fixed points in the plane.
- A halting configuration represents the result of a computation after the computer program completes its computation.
- f is structurally stable if all dynamical systems g that are close to f via some metric, then f and g are topologically conjugate.

MAIN IDEA: If Turing machine $\mathcal{M}$ is structurally stable, then SMALL CHANGES to the computer program $\mathcal{M}$ WILL NOT CHANGE WHAT $\mathcal{M}$ COMPUTES!

Can We Find Standard Computation That Is Stable?

- A Universal Turing Machine is basically a compiler or interpreter (C, JAVA, Lisp, Python, ...).
- In our paper, a Universal Turing machine (UTM) is provided and it is shown that this UTM is structurally unstable.
- This means that given a computer program P there are other programs arbitrarily close that exhibit different dynamics.
- This means some of the nearby computer programs do not perform the same computation as program P.

All Register Machine Compilers might be Unstable

- These initial results suggest that all compilers for C, Python, etc. might be structurally unstable.
- Math theory translates to computer programs executing on register machines are inherently susceptible to malware vulnerabilities.
- Caveat: instability was only shown for two metrics and one UTM encoding.

Research Questions for Conventional Machines

- Are there general mathematical conditions when a Turing machine (register machine) is structurally unstable?

- Do useful metrics exist on the space of Turing machines (computer programs)?

WITI: IF stable conditions exist, THEN WE CAN DESIGN

robust digital computer programs with our know-how.

Research Direction for Unconventional Machines

IF stable conditions DO NOT EXIST, THEN WE CAN

- BUILD machines that simultaneously execute instructions
- BUILD machines that can repair their instructions.

WITI: The program purpose can be made stable.

Computation is resistant to sabotage of instructions.

Active Element Machine[5]

All Elements compute simultaneously.

Elements fire and send pulses along Connections.

Connections specify the pulses sent between elements.

An element E fires at time s if:

the sum of the input pulses to E is greater than E's threshold

AND E's refractory period r has expired;

that is, $s \geq r + l$ where l is the most recent firing time of E.

[5] M.S. Fiske. The Active Element Machine. Proceedings of Computational Intelligence. Autonomous Systems. Volume **391**. Springer, 2011, pp. 69-96.

Connections

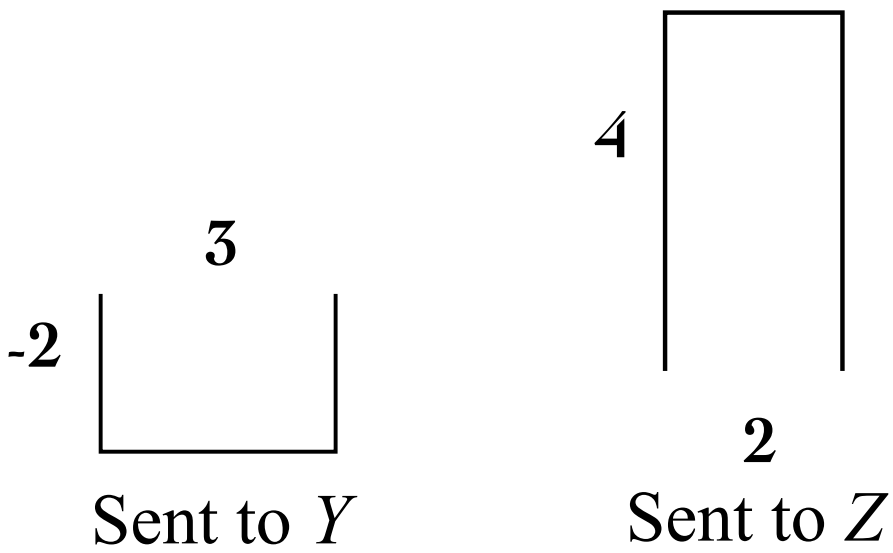
(connection (time 4) (from E) (to Y) (amp -2) (width 3) (delay 5)

(connection (time 4) (from E) (to Z) (amp 4) (width 2) (delay 3)

If element E fires at time 4, then

A pulse of time width 3 and amplitude -2 reaches Y at time 9.

A pulse of time width 2 and amplitude 4 reaches Z at time 7.



Meta Command

Keyword *clock* evaluates to the current time of the active element machine clock.

(meta (name E) (window b e) (C (args clock)))

If active element E fires at time s in window $[b, e]$,

where $b \leq s \leq e$, then command (C clock) executes at time s.

If there is no window and if E fires at any time s,

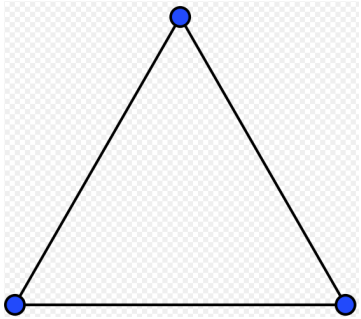
then (C s) executes at time s.

Searching for Complete Graphs

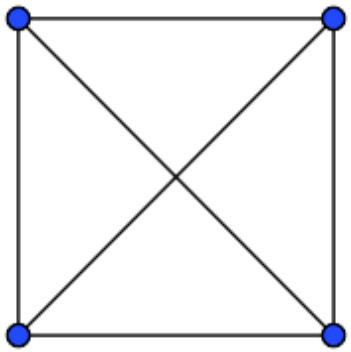
A complete graph K_n on n vertices: every vertex pair has one edge.

K_2

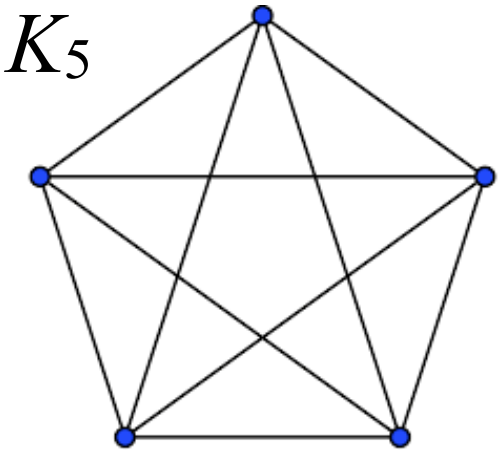
K_3



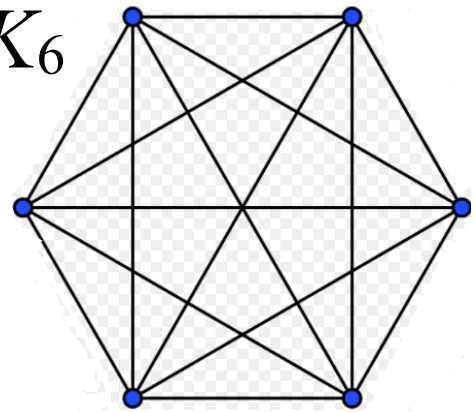
K_4



K_5

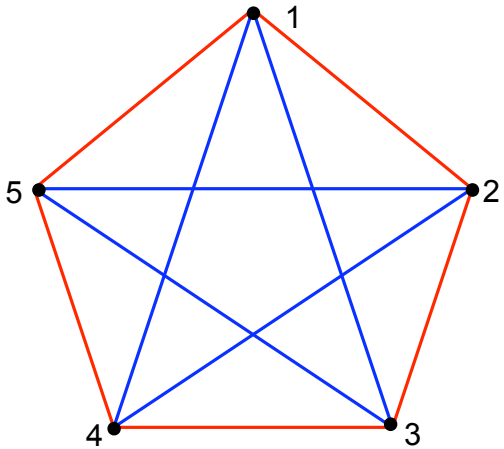


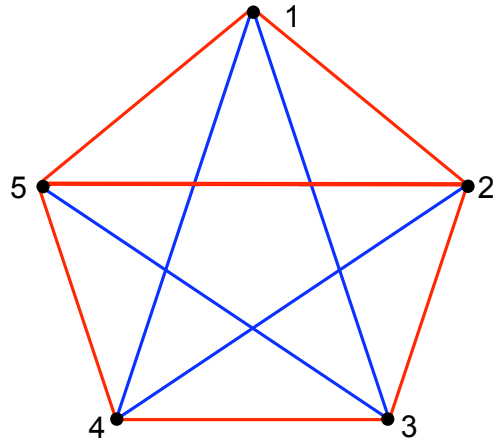
K_6



Ramsey Numbers

Each edge of K_n is colored red or blue.





Ramsey number $r(j, l)$ is the least integer n such that:

There is at least one complete subgraph K_j with only red edges

OR

There is at least one complete subgraph K_l with only blue edges.

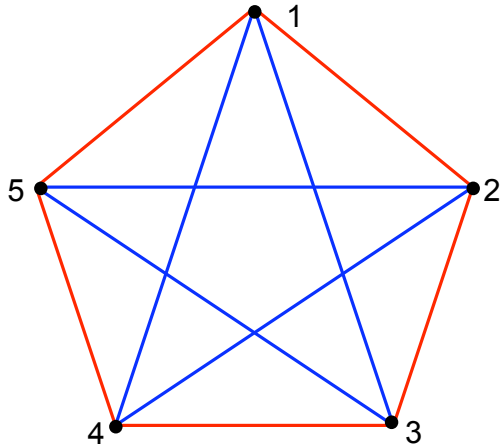
Determining $r(m, m)$ is NP-hard. $r(5, 5)$ is unknown.

AEM Program Determines $r(3,3) > 5$

Red edges: $\{1,2\}, \{2,3\}, \{3,4\}, \{4,5\}, \{1,5\}$

Blue edges: $\{1,3\}, \{1,4\}, \{2,4\}, \{2,5\}, \{3,5\}$

Triangles: $\{1,2,3\}, \{1,2,4\}, \{1,2,5\}, \{1,3,4\}, \{1,3,5\},$
$\{1,4,5\}, \{2,3,4\}, \{2,3,5\}, \{2,4,5\}, \{3,4,5\}$



1. Red edge Element commands

(element (time 0) (name R_12) (threshold 1) (refractory 1) (last -1))

(element (time 0) (name R_23) (threshold 1) (refractory 1) (last -1))

(element (time 0) (name R_34) (threshold 1) (refractory 1) (last -1))

(element (time 0) (name R_45) (threshold 1) (refractory 1) (last -1))

(element (time 0) (name R_15) (threshold 1) (refractory 1) (last -1))

AEM Program Determines $r(3,3) > 5$

2. Blue edge Element commands

(element (time 0) (name B_13) (threshold 1) (refractory 1) (last -1))

(element (time 0) (name B_14) (threshold 1) (refractory 1) (last -1))

(element (time 0) (name B_24) (threshold 1) (refractory 1) (last -1))

(element (time 0) (name B_25) (threshold 1) (refractory 1) (last -1))

(element (time 0) (name B_35) (threshold 1) (refractory 1) (last -1))

3. Fire R_jk if edge $\{j,k\}$ is red.

(fire (time 0) (name R_12)

(fire (time 0) (name R_23)

(fire (time 0) (name R_34)

(fire (time 0) (name R_45)

(fire (time 0) (name R_15)

4. Fire B_jk if edge $\{j,k\}$ is blue.

(fire (time 0) (name B_13)

(fire (time 0) (name B_14)

(fire (time 0) (name B_24)

(fire (time 0) (name B_25)

(fire (time 0) (name B_35)

AEM Program Determines $r(3,3) > 5$

5. For each edge $\{j,k\}$,

(meta (name R_jk) (window 0 1)
(connection (time 0) (from R_jk) (to R_jk) (amp 2) (width 1) (delay 1)))

(meta (name B_jk) (window 0 1)
(connection (time 0) (from B_jk) (to B_jk) (amp 2) (width 1) (delay 1)))

6. For each $\{i,j,k\}$, compute if blue triangle on vertices $\{i,j,k\}$.

(connection (time 0) (from B_ij) (to B_ijk) (amp 2) (width 1) (delay 1)))
(connection (time 0) (from B_jk) (to B_ijk) (amp 2) (width 1) (delay 1)))
(connection (time 0) (from B_ik) (to B_ijk) (amp 2) (width 1) (delay 1)))

7. For each $\{i,j,k\}$, compute if red triangle on vertices $\{i,j,k\}$.

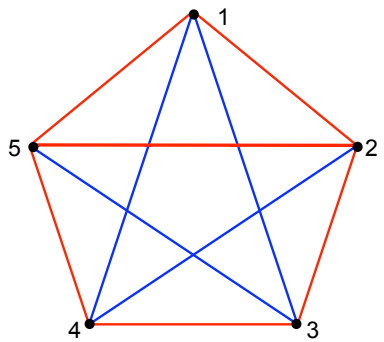
(connection (time 0) (from R_ij) (to R_ijk) (amp 2) (width 1) (delay 1)))
(connection (time 0) (from R_jk) (to R_ijk) (amp 2) (width 1) (delay 1)))
(connection (time 0) (from R_ik) (to R_ijk) (amp 2) (width 1) (delay 1)))

8. For each vertex set $\{i,j,k\}$, create red and blue elements.

(element (time 0) (name R_ijk) (threshold 5) (refractory 1) (last -1))
(element (time 0) (name B_ijk) (threshold 5) (refractory 1) (last -1))

AEM Program Sabotage and Repair

```
Michaels-MacBook-Air:AEM_K5_repair michael_fiske$ diff K5_1_red_triangle.aem K5_1_red_triangle_sabotage.aem
64,65c64,66
< (meta (name R_12) (window 0 1)
<       (connection (time 0) (from R_12) (to R_12) (amp 2) (width 1-dT) (delay 1)))
---
> ;;  SABOTAGE (meta (name R_12)
> ;;  (meta (name R_12) (window 0 1)
> ;;        (connection (time 0) (from R_12) (to R_12) (amp 2) (width 1-dT) (delay 1)))
```




```
Michaels-MacBook-Air:AEM_K5_repair michael_fiske$ diff K5_1_red_triangle.aem repair_K5_1_red_triangle.aem
64,65c94,96
< (meta (name R_12) (window 0 1)
<       (connection (time 0) (from R_12) (to R_12) (amp 2) (width 1-dT) (delay 1)))
---
> ;; SABOTAGE  (meta (name R_12))
> ;; (meta (name R_12) (window 0 1)
> ;;      (connection (time 0) (from R_12) (to R_12) (amp 2) (width 1-dT) (delay 1)))
```